Software Risk Management through Independent Verification and Validation

John R. Callahan[1]
Tong C. Zhou
Ralph Wood
Department of Statistics & Computer Science
Concurrent Engineering Research Center
West Virginia University

Abstract

Software project managers need tools to estimate and track project goals in a continuous fashion before, during, and after development of a system. In addition, they need an ability to compare the current project status with past project profiles to validate management intuition, identify problems, and then direct appropriate resources to the sources of problems. This paper describes a measurement-based approach to calculating the risk inherent in meeting project goals that leverages past project metrics and existing estimation and tracking models. We introduce the IV&V Goal/Questions/Metrics model, explain its use in the software development life cycle, and describe our attempts to validate the model through the reverse engineering of existing projects.

1 Introduction

Managers of large, complex software projects often rely on independent contractors to verify and validate (V&V) the computer software produced by a separate development contractor. An independent V&V (IV&V) contractor helps identify, manage, and reduce the potential *risk of failures* to meet intended requirements in a software project at all phases of development. While some level of risk will always remain in a project, risk can be reduced if errors and other discrepancies are found as early as possible in the software development life cycle. Many studies have shown that undetected errors in a project will increase the likelihood of failures in later life cycle phases when the cost to fix them increases by orders of magnitude.

IV&V efforts are highly effective in early life cycle phases [1] if they can successfully predict the likelihood of problems based on an analysis of the current state of a project. It is difficult, however, to make such predictions with provable accuracy and show correlation between development activities and problems that arise in later life cycle phases. Formal software development models can provide some insight based on quantified analysis of *past* software development efforts [2,3]. While such formal models are imperfect guides to future efforts, they are far more likely to predict problems than informal methods.

We have developed an approach called the IV&V Goal/Question/Metric method (IGQM) that allows IV&V managers to monitor the level of risk in a software development project. Using IGQM, managers can use past projects as "yardsticks" against which to measure present projects. They can also assess the potential impact of their decisions about resource allocations, schedules, costs, and tradeoffs

[1]This work is supported by DARPA Grant MDA 972-91-J-102 under the DARPA Initiative for Concurrent Engineering (DICE) program, NASA Grant NAG 5-2129 and NASA Cooperative Agreement NCCW-0040 under the NASA Independent Software Verification and Validation (IV&V) Facility, Fairmont, WV.

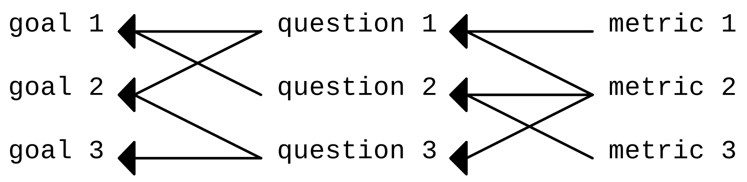

Figure 1: The Goal/Question/Metric (GQM) model

during execution of the development effort. The IGQM method provides *continuous* reporting of the status of a project in terms of what areas are at risk of failure. The method represents a formal interface between the IV&V contractor, the software development contractor, and the customer. It summarizes the analysis work performed by the IV&V contractor in terms of what project goals are at risk of failure and allows managers to make informed decisions about why problems are occurring.

This paper discusses the IGQM model and its use in an automated support environment [4]. Unlike existing metric-based models, our approach does not emphasize any specific set of metrics or functions for assessing risk. The model allows for use of other assessment models. The IGQM model is used to collect and summarize the metrics and relate them directly to project goals. Although our approach to IV&V relies on metrics from past projects as baselines, the model can be "primed" with informal estimates or external project databases. Results from pilot projects are then used as feedback to provide continuous improvement to the model itself in order to improve our predictive accuracy.

The IGQM model can incorporate several existing software estimation and tracking methods. These include the COCOMO method [2] and Software Equation [3] for estimating cost, size, and effort. We describe our attempts to validate our approach by using these methods to reverse engineering past projects to determine if identifying risk sources early in the life cycle could have helped prevent later problems.

The IGQM model is embedded in an automated support environment for software IV&V [4] that allows continuous analysis of a project's status. IV&V is viewed as a complementary process to the software development process [5] and it is responsible for continuous assessment of the development process. As a software development process progresses, events are triggered in the IV&V process. The IV&V team must analyze changes in the development process and publish its findings to the customer in the form of an IV&V report. This report is generated using the IGQM method by a tool that is integrated into a CASE environment. The reporting tool collects and summarizes analysis results (i.e., metrics) from other IV&V CASE tools in an incremental fashion. When a change occurs in the development process, the project measurements and risk are updated incrementally like values and formulas in a spreadsheet. The risk impact of each change is assessed immediately relative to the project goals. This paper does not discuss the details of the automated support environment, but focuses on the IGQM model around which the environment is organized.

2 Approach

The IGQM approach to software IV&V focuses on the quantification, identification, management, and reduction of risk in software development projects based on objective metrics taken during the software development life cycle. Metrics include process measures (i.e., whether or not a particular procedure been performed at this phase) as well as artifact measures (i.e., quantitative measurements of documents, code, tests, and other products). The IGQM tool formally defines the impact of such measures on the failure or success in meeting project goals.

Our approach is based on the Goal-Question-Metric (GQM) model [6] augmented with risk analysis [7]. The GQM model depicted in Figure 1 allows managers to explicitly describe a

Project	Confidences				Certainty	Uncertainty	Importance	Risk
	Q1	Q2	Q3	Q4				
G1	1.00	0.36	0.77	0.00	0.45	0.55	0.80	0.440
G2	1.00	0.36	0.77	0.00	0.78	0.22	0.30	0.066
G3	1.00	0.36	0.77	0.00	1.00	0.00	0.90	0.00
G4	1.00	0.36	0.77	0.00	0.04	0.96	0.10	0.096

Table 1: Computing goal risks based on question confidence probabilities

Questions	M1	M2	M3	M4	confidences
Q1	34	11	88	99	1.00
Q2	34	11	88	99	0.36
Q3	34	11	88	99	0.77
Q4	34	11	88	99	0.00

Table 2: Computing question confidence probabilities based on project metrics

project in terms of a set of goals so that the development team has more precise knowledge about the intent of the customer. A project must completely satisfy a set of goals to be implemented successfully. Goals include requirements but are much broader and can include ambiguous statements like "the system must be highly reliable." Each goal is satisfied by answering a set of related questions. The questions define the features needed to satisfy a particular goal. Questions are answered true or false, but can be parameterized with limits, e.g., "does the system have a 10,000 hour mean time between failures?" Each question is answered based on a set of quantifiable project metrics. A metric might be "lines of code" or "estimated mean time between failures" or any other discrete value. The GQM approach is used as a dialogue between customers and development organizations for agreeing on the details of a project. In this fashion, it should be clear to the developer exactly what is expected of the final product and the criteria for its acceptance.

We have augmented the GQM model to compute the risk of failure in a project to satisfy the intended goals. The risk of failing to satisfy the goal is defined as the uncertainty of reaching that goal multiplied by the importance of that goal. Table 1 shows a list of goals, their importances, certainty, uncertainty, and risks for an example project. The goals G1,...,G4 might be

- Low cost
- Medium effort
- Use of prototyping
- High reliability

The questions related to each goal in the IGQM model will determine exactly what is meant by each goal. The risk values associated with each goal should change during the software development life cycle. If we keep track of the risk at each step in the development process, we can identify high-risk goals and ensure that the overall risk is non-increasing over time, i.e., while risk may increase at any step, the overall risk trend is decreasing.

2.1 Risk associated with each goal

To calculate the risk associated with each goal, the importance of the goal is specified explicitly by the manager, but its certainty is computed from answers to related questions in the GQM model. For each goal-questions group, we employ a set of *certainty functions G* at each step of the development life cycle defined as

$$g_{i,tp} = G_{i,tp}(Q_{x,tq}, Q_{y,tr}, \ldots)$$

where $g_{i,tp} \in [0 \ldots 1]$ for the i^{th} goal at the process step t_p and each $Q_{x,tn}$ is the probabilistic confidence answering question x as true at process step t_n. Thus, the certainty of satisfying each goal changes at each step in the software development process. The certainty

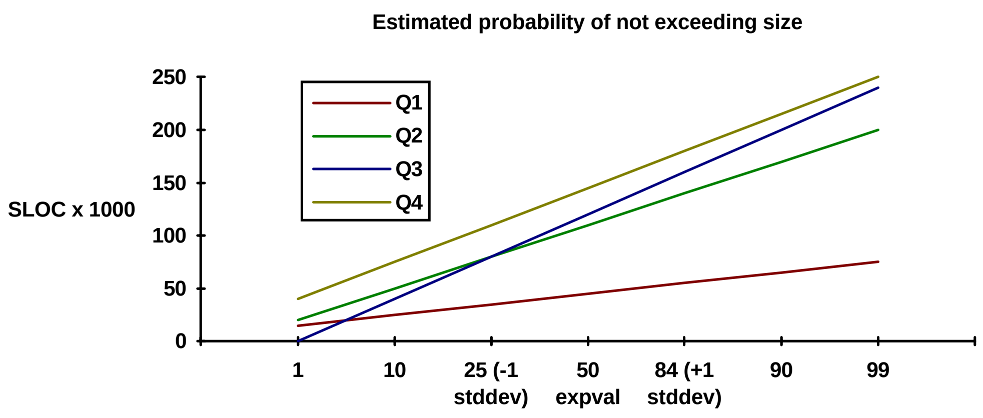

Figure 2: Confidence functions for estimated SLOC in early life cycle phases

functions may be based on the baselines of past projects or on the results of simulated models. In either case, the results of certainty functions are added to the baseline for use in future projects.

2.2 Confidence in answers to questions

Here is where existing estimation and tracking methods fit into the IGQM model. Each question can be answered true with a characteristic probability called its *confidence*. A false answer has a confidence value of zero. The confidence of answering a question is determined by a unique function based on collected project metrics. For each question-metrics group, we employ a set of *confidence functions* Q defined as

$$q_{x,tp} = Q_{x,tp}(M_{a,tq,se}, M_{b,tr,sf}, \ldots)$$

where $q_{x,tp} \in [0 \ldots 1]$ for question x at the process step t_p where each $M_{a,tz,se}$ is a metric a at step t_z provided by source s_e. Table 2 shows a question and its related metrics from which a confidence function is defined. All metric values are the same relative to each question, but the confidence functions are defined uniquely for each question and process step. Metrics that are unknown at process steps can still be used because the lack of knowledge contributes to the risk calculation. Unknown measure decrease confidence in answering questions and in turn decreases the certainty of satisfying a goal.

3 Predictive Functions

The characteristic certainty and confidence functions associated with goals and questions can be based on many existing methods that have evolved from experiences on large numbers of actual projects. The IGQM model simply tries to relate the calculation of risk to the analysis these methods provide in order to help identify areas of a project that need attention and allow managers to trace problems to their sources.

For example, several methods exist for estimating the eventual number of source lines of code (SLOC) in a project [3]. Early estimates of SLOC will be very inaccurate, but we can assess the probability of the correctness of our estimate. Consider the goal of "Small Program" in which the related questions are:

1. Are there less than 100 requirements?
2. Are there less than 50 function points?
3. Are there less than 50 modules?
4. Are there less than 10,000 SLOC?

In this example, question 4 might given the most weight in ultimately determining the acceptance criteria. However, in the early stages of a project, we can only answer question 1 with a large degree of confidence, but the answer to this question will not have a large impact of increasing the certainty of meeting the goal according to our weighting. Figure 2 shows a risk profile for the different questions at this stage of development. The relatively higher slopes of the other questions illustrates a greater degree of uncertainty.

The weighting of each question confidence measure in determining goal certainty will change during the lifetime of the project, i.e., the slopes will decrease and different measures will play larger roles. Eventually, confidence functions may get better with more experience and a broader database of actual projects. This will also decrease the uncertainty.

Estimating functions are highly domain dependent. This is why it is important for each organization to institute measurement programs to improve the effectiveness of their predictions. The IGQM model can be primed with hand-picked estimate or those from external projects, but these initial estimates will be highly inaccurate. Only with time can an organization build confidence in their predictive models. Of course, changes in personnel and the need to tackle new projects can invalidate previous experience, but

In the case of SLOC, we can determine the probability of the eventual number of lines of code exceeding our estimate. Likewise, many methods exist for cost, size, error, and effort estimation. Whereas many of these techniques are only used early in a project to construct a proposal or plan, our approach allows managers to track actual measurements and compare them with estimates. As a project evolves, a manager can gain greater confidence in the estimates as they change dynamically based on actual performance.

4 Discussion

In Table 1, we can see that goal G1 is the only goal with significant associated risk. If the confidence and certainty functions are based on methods that leverage past project data, the risk associated with G1 at this process step might say something like "44% of the projects at this step with a similar goal-questions profile failed to successfully satisfy this goal at time of delivery." The interpretation is based on the characteristic confidence and certainty functions related to each goal and question respectively.

Creating the certainty and confidence functions is not easy. They are based on profiles of past projects, contain coefficients that are specific to each environment or project, and must be primed initially with estimates or data from external projects. By mapping our approach to current software development and V&V practices, we "reverse" engineered these estimates from informal measures on past projects. Even though some information was not available on these projects, they were adequate enough to provide working estimates. In one case we wanted to verify the intuition of V&V personnel who noted problems with the delivery schedule of project milestones. In their expert opinion, the schedule was too short. Our model, based on existing methods such as COCOMO, confirmed that the intuition was correct.

In the next sections, we show how traditional V&V activities can be mapped to our model. Specifically, we relate process management and testing to see how they contribute to project measurements and at what stages of the life cycle. Based on this mapping, we can assess the relative effectiveness of these traditional approaches in controlling software projects. Process management, for example, ensures that the software development team follows all process steps (e.g., DOD 2167A) and follows up on all discrepancy reports and anomalies. It monitors that the proper artifacts (i.e., documents and code) are produced on time and in their proper order. Testing, on the other hand, is usually associated with code level validation of the end-product system in a simulated environment. While it is widely believed that both of these approaches help reduce project risk, they have serious limitations in many projects, especially in large, complex systems with volatile requirements. It is possible that expensive and catastrophic errors may go undetected using traditional approaches. We show that according to our reverse engineered projects, late life cycle testing may find some errors but it is often too late to fix them. This fact shows up not as an increase in risk towards the end of the project, but as an inability of existing techniques to keep the risk trend non-increasing and within nominal limits.

Process management and testing alone are inadequate means to manage risk in large, complex projects.

4.1 Process

First, an IV&V team can check to make sure the software development process is followed at all steps. The goal of this task is to reduce risk by ensuring that a process is followed that increases the probability of success. The reasoning behind this task is informal: if a past task was successful using a process then each step must be repeated to guarantee success in other projects.

We can cast current software development practices into the IGQM risk model by asking specific questions about process steps accomplished. The metrics are Boolean values that help answer questions at each step regarding whether or not a procedure has been performed. In this fashion, the IGQM approach subsumes these current "checklist" methods and provides a metrics-based environment for formally validating whether or not generic assumptions about process effectiveness are true. Process tracking by the IV&V team is necessary but insufficient to ensure risk reduction in the development project.

4.2 Testing

Software testing has been a major focus of IV&V efforts, but testing is expensive and has severe limitations. Traditional testing cannot find many problems or finds problems too late in the software development life cycle where they are too costly to fix. In the IGQM model, the results of tests can be viewed as metrics (e.g., pass-fail). From this metrics-based perspective, early analysis of requirements and design can also be viewed as "tests" but the test results are viewed with less confidence than concrete tests at later stages of the development life cycle. In addition, the tests can be directly associated with requirements or project goals in the IGQM model. In this case, the existence of a test is important for tracability. We used this approach to model traditional testing in the IGQM model and showed that late testing reduces risk, but that the risk trend is already too high at later phases for testing to have any significant effect.

Traditional testing does not permit early detection of problems and it is often impossible to exercise a system with a battery of tests that completely characterize the operational environment. If major problems occur, it is often too late and expensive to fix them. As a result, the software might experience traumatic failure, the project is scrapped, must be redone, or the customer is left dissatisfied with a partially functional system. If the customer had access to effective, predictive estimate earlier in the development process, expectations might be more realistic and the intentions better defined with the development team.

5 Implementation

We have implemented the IGQM model in a tool for use by IV&V practitioners. The tool, called ADMIT (A Distributed Metrics Integration Tool), is implemented in Tk/Tcl under the X Windows system and the UNIX operating system. The tool primarily consists of three list boxes of goals, questions, and metrics and a multi-graph widget that shows the cumulative risk for the project, per goal, question confidences, and metric values.

Metrics come from many sources in the distributed environment. Some come from shared files and databases (e.g., the Network File System (NFS), Oracle). When the files change, the tool read the file, updates the measure, and recalculates the associated confidence and certainty functions. Metric may also be source directly from CASE tools using remote procedure call in which the ADMIT tool acts as the server. We are continuing to evolve our implementation as we integrate other sources of metrics and techniques for collecting them.

6 Summary

Our approach depends on an intense metrics collection and archival capability to provide high levels of confidence in IV&V predictions. It also depends on the continuous evolution of the predictive certainty and confidence functions. While our approach does not eliminate risk from a project, it does formalize the risk identification, management, and reduction. It makes risk management the explicit objective of the IV&V process in order to deliver effective results to the customer. Moreover, the confidence of predictions can be increased as our baseline grows with each project. For well-defined application domains, we expect this approach will have most value based on extrapolating experiences with the IGQM model in practice.

While a statistical risk model of IV&V does not guarantee success, it represents a significant improvement over existing practices that deliver dubious value to the IV&V customer and may unknowingly harm software development efforts with needless paperwork. During the course of our research, we continue to investigate (1) effective process models; (2) specific and useful metrics and their correlation within the process; and (3) continuous improvement of certainty and confidence functions associated with the process.

7 References

[1] The Cost-Effectiveness of Independent Software Verification and Validation, NASA Jet Propulsion Laboratory, 1985.

[2] Boehm, B., *Software Engineering Economics*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[3] Putnam, L. and W. Myers, *Measures for Excellence: Reliable Software on Time, within Budget*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.

[4] Karinthi, R., S. Kankanahalli, S. Reddy, C. Cascaval, W. Jackson, S. Venkatraman, H. Zheng, Collaborative Environment for Independent Verification and Validation of Software, *In Proceedings of the Third IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, April 17-19, 1994, Morgantown, WV.

[5] Lewis, R., Independent Verification and Validation: A Life Cycle Engineering Process for Quality Software, John Wiley & Sons, New York, 1992.

[6] Basili, V., Applying the Goal/Question/Metric Paradigm in the Experience Factory, in *Software Quality Assurance: A Worldwide Perspective*, Chapman & Hall Publishers, 1994.

[7] Cardenas-Garcia, S. and M. Zelkowitz, A Management Tool for Evaluation of Software Designs, *IEEE Transactions on Software Engineering*, Volume 17, Number 9, September 1991.

8 Biographies

John R. Callahan is an assistant professor of computer science in the Department of Statistics and Computer Science at West Virginia University and a research faculty member at the Concurrent Engineering Research Center (CERC) in Morgantown, West Virginia. He received his Ph.D. from the University of Maryland College Park in 1993 in software engineering and is currently working on research in independent verification and validation of computer software. He has worked for Xerox Corporation of Tyson's Corner, Virginia and Palo Alto, California as well as NASA Goddard Space Flight Center and the Department of Defense. Dr. Callahan serves as the NASA Research Associate at the NASA IV&V Facility in Fairmont, West Virginia. The Fairmont facility houses IV&V contract work for the Mission To Planet Earth and Space Station projects.

Tong C. Zhou is a graduate student in the Department of Statistics and Computer Science at West Virginia University and a researcher at the Concurrent Engineering Research Center (CERC) in Morgantown, West Virginia. Her interests include formal methods, risk analysis, and automated test generation.

Ralph Wood is a visiting scientist at the Concurrent Engineering Research Center (CERC) in Morgantown, West Virginia. Dr. Wood is a former senior engineering scientist for General Electric Research and Development. His interests include risk management and cost/schedule estimation.